Return and Refund Management System (RRMS)

Functional Requirement Document (FRD)

Table of Contents

1. Document Overview:

Title: Functional Requirements Document – Return & Refund Management System (RRMS)
Project Name: Return & Refund Management System (RRMS)
Author: Namrutha H
Version: 1.0
Date Created: October 22, 2025
Status: Draft

2. Purpose:

The purpose of this Functional Requirements Document (FRD) is to define the detailed functional specifications for the Return and Refund Management System (RRMS) for Shop Swift, a mid-sized e-commerce platform. This document translates the business needs outlined in the BRD into clear, actionable system functionalities. The RRMS is designed to address issues such as inconsistent return processes, delayed refunds, and limited customer support by implementing automation and system-driven workflows. This FRD serves as a guide for the design, development, and validation of the system features.

3. Scope:

This Functional Requirements Document (FRD) defines the in-scope and out-of-scope functionalities for the Return and Refund Management System (RRMS) to be implemented on the Shop Swift e-commerce platform.

3.1 In-Scope

- Customer-initiated product returns via the online platform
- Automated eligibility checks for returns
- Real-time tracking of returns and refunds
- Automated refund processing through payment gateway integration

3.2 Out-of-Scope

- Physical return logistics (e.g., packaging, pickup, and shipping)
- Inventory and restocking management

4. Functional Requirements

ID	Requirement Description
FR-001	The system shall allow customers to initiate return requests from their order history.
FR-002	The system shall validate return eligibility based on configurable rules (e.g., return window, product category).
FR-003	The customer shall be able to select a reason for return and optionally upload supporting images.

FR-004	The system shall provide real-time status updates for each return and refund request.
FR-005	The system shall notify customers via email or SMS at key stages (e.g., request received, approved, refunded).
FR-006	The system shall integrate with payment gateways to automate refund processing once a return is approved.
FR-007	The system shall provide an internal dashboard for support and operations teams to manage and track returns.
FR-008	The system shall log all return and refund activities for audit and reporting purposes.
FR-009	The system shall allow admins to override, approve, or reject return requests manually.

5. Use Case Descriptions

5.1 Initiate Return Request	
Field	**Description**
Use Case ID	UC-001
Use Case Name	Initiate Return Request
Actor(s)	Customer
Preconditions	Customer is logged into their Shop Swift account and has an eligible order.
Main Flow	1. Customer navigates to order history.2. Selects item to return.3. Clicks "Return Item".4. Chooses return reason and uploads image (optional).5. Submits return request.
Postconditions	Return request is submitted and confirmation is shown. Customer is notified via email/SMS.
Exceptions	- Product is outside return window → Error message shown. - Invalid image format → Validation error.

5.2 Track Return status	
Field	**Description**
Use Case ID	UC-002
Use Case Name	Track Return Status
Actor(s)	Customer
Preconditions	A return request has been submitted.
Main Flow	1. Customer logs in.2. Navigates to return history or order details.3. Views return/refund status (e.g., Pending, Approved, Refunded).
Postconditions	Real-time status is displayed accurately.

Exceptions	- Status not updated → "In Progress" message shown with last known update.

5.3 Process Refund (System + Admin)	
Field	**Description**
Use Case ID	UC-003
Use Case Name	Process Refund
Actor(s)	System, Finance Admin
Preconditions	Return request has been approved.
Main Flow	1. System triggers refund request.2. Payment gateway processes refund.3. Confirmation sent to customer.4. Finance team monitors/refunds manually if failure occurs.
Postconditions	Refund is issued and logged. Status is updated to "Refunded".
Exceptions	- Payment gateway fails → Admin notified, manual processing initiated.

6. User Interface (UI) Requirements

6.1 Return Request Page (Customer UI)	
Purpose: Allow customers to submit a return request from their order history.	
Element	**Description**
Return Button	Visible next to eligible products in order history
Reason for Return	Dropdown list (e.g., "Damaged", "Wrong item", "No longer needed")
Upload Image	Optional file upload (JPEG, PNG)
Submit Button	Submits the return request
Validation Rules	Required fields: Reason for Return; Image (optional); Only for eligible items
Success Message	Confirmation shown: "Your return request has been submitted successfully."

6.2 Return Status Tracking Page	
Purpose: Show customers real-time updates on return and refund status.	
Element	**Description**
Product Name	Displays product being returned
Return ID	Unique identifier for return
Current Status	Status tracker (e.g., Pending, Approved, Refunded)
Estimated Refund Date	Displays expected date of refund

Contact Support Link	For issues or escalation

6.3 Admin Dashboard (Internal UI)	
Purpose: Allow internal teams (support/operations) to view and manage return requests.	
Element	**Description**
Return Requests Table	List of return requests with filters (Date, Status, Customer)
Action Buttons	Approve / Reject / Mark as Refunded
Search Function	Search by Order ID, Return ID, or Customer Email
Return Details Panel	Shows full request info (reason, images, history)
Notification Toggle	Enable/disable customer notifications per return

7. Integration Requirements

7.1 Payment Gateway Integration	
Purpose	**To automate refund processing directly to the customer's original payment method.**
System	Razor pay / Stripe / PayPal (based on actual system used)
Data Exchanged	Refund amount, payment method, transaction ID, refund status
Trigger Point	After return request is approved
Expected Response	Success/failure confirmation with transaction reference
Failure Handling	Notify admin and allow manual refund override

7.2 Logistics / Courier Service Integration	
Purpose	**To track return pickup and delivery progress.**
System	Shiprocket / Delhivery / Custom API
Data Exchanged	Pickup request, tracking ID, status updates
Trigger Point	When return is approved and scheduled for pickup
Expected Response	Pickup scheduled, picked up, in transit, delivered
Failure Handling	Fallback to manual update by support team

7.3 Notification System (Email/SMS Service)	
Purpose	**To send status updates to customers during the return and refund process.**
System	SendGrid / Twilio / SMS Gateway
Data Exchanged	Customer email/phone, message content, delivery status

Trigger Points	On return initiation, approval, refund issued, exception
Expected Response	Delivery confirmation

8. Assumptions and Constraints

Assumptions

These are conditions believed to be true for the purpose of designing and implementing the system:

- All customers have internet access and can log in to initiate returns.
- Return eligibility rules (e.g., 7-day return window, product condition) are predefined and configurable.
- Sellers will follow the standard return policy enforced by the platform.
- The payment gateway used supports refund APIs and is available during business hours.
- Logistics partners provide real-time tracking APIs or status update webhooks.
- Customer support and admin teams are trained to use the internal dashboard.

Constraints

These are limitations or restrictions that may affect the design, development, or performance of the system:

- Refund processing depends on third-party payment gateway uptime and APIs.
- Real-time tracking may be limited by the logistics provider's API update frequency.
- The return request feature is only available for orders placed online (not in-store, if applicable).
- UI and features must be compatible with both desktop and mobile views.
- The system must comply with relevant data protection and privacy laws (e.g., GDPR, if applicable).
- Any major changes to return policies will require system reconfiguration or updates.